–TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM **TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KỲ**

**XỬ LÝ DỮ LIỆU LỚN**

*Người thực hiện*:**Nguyễn Trung Nam – 519H0321**

Lớp **: 19H50302**

Khoá **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM **TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KỲ**

**XỬ LÝ DỮ LIỆU LỚN**

*Người thực hiện*:**Nguyễn Trung Nam – 519H0321**

Lớp **: 19H50302**

Khoá **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

1

**PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

1

|  |  |  |  |
| --- | --- | --- | --- |
| **MỤC LỤC**  ***Trang***  [Thu thập dữ liệu](#_heading=h.go062s2kn804)  [1.1. Cào dữ liệu và mô tả cấu trúc](#_heading=h.fpput7l3mnz)  [1.2. Thao tác với dữ liệu](#_heading=h.s6i7yaaud03n)  [1.2.2. Hãy tìm trong biến 'result’ vừa rồi các dữ liệu có chứa nhãn "small" với class là "author" và in kết quả ra màn hình?](#_heading=h.nxmrrxso3rn5)  [1.2.3. Hãy viết hàm tacgiaLink() để lấy nội dung của mỗi tác giả. Với mỗi tác giả in ra màn hình các nội dung tên tác giả,đường link của tác giả, ngày tháng năm sinh và câu nói nổi tiếng của tác giả.](#_heading=h.j4vbtjdvahek)  [1.2.4. Hãy lưu kết quả ở câu c vào file Quote.csv tương ứng, với mỗi tác giả là 1 dòng dữ liệu. Bạn được yêu cầu thu thập ít nhất 40 câu nói nổi tiếng từ trang web trên một cách tự động theo code của các ý trên?](#_heading=h.ezdmdz5dgmau)  [Khai phá dữ liệu](#_heading=h.wfpz5yip4bal)  [2.1 Xử lý dữ liệu](#_heading=h.ioruchk6b4e4)  [2.2.1 Một số giá trị của dữ liệu Trường ngày sinh chưa có, bạn hãy đề xuất cách điền?](#_heading=h.iuh0s07gqjc4)  [2.2.2 Bạn hãy thêm vào Trường Tuoi (Tuổi) và đề xuất cách điền tuổi của các tác giả?](#_heading=h.8pd1lkrmtb2p)  [2.2 Khám phá dữ liệu](#_heading=h.slmd6sj4jdct)  [2.2.1 Thống kê về tác giả và câu nói nổi tiếng có trong bộ dữ liệu](#_heading=h.76x7t67s7kub)  [2.2.2 Thống kê về năm sinh và độ tuổi của các tác giả](#_heading=h.jf5a8xm2ueap)  [2.2.3 Thống kê về các câu nói nổi tiếng như: câu dài nhất, ngắn nhất, số từ](#_heading=h.70bfmmfs64mf)  [2.3. Trích xuất đặc trưng- Feature Extraction](#_heading=h.ynb81s26oe87)  [2.4. Suy luận](#_heading=h.gnkt2prnswr0)  [2.4.1 Hãy dự đoán tên của người nổi tiếng theo câu nói dựa trên các đặc trưng bạn trích xuất ở trên và đánh giá trên bộ dữ liệu đã cho với tỉ lệ Train/Test và các độ đo phù hợp?](#_heading=h.j25k66h22b1z)  [2.4.2 Hãy đề xuất cách tính độ tương đồng phong cách nói giữa các tác giả và tìm ra các tác giả có phong cách nói tương đồng nhau nhất?](#_heading=h.3iat4zdio0e8)  [TÀI LIỆU THAM KHẢO](#_heading=h.tniauen8tgig) | | |  |

2

## Thu thập dữ liệu

### 1.1. Cào dữ liệu và mô tả cấu trúc

Cào dữ liệu và lưu kết quả vào file kq.txt :

Code:

from urllib.request import urlopen

from bs4 import BeautifulSoup

url ="http://quotes.toscrape.com/"

page = urlopen(url)

soup = BeautifulSoup(page, 'html.parser')

# Lưu vào file tên kq.txt

with open('kq.txt', 'w') as f:

f.write(soup.prettify()) #soup.prettify() turn soup to Unicode string

# df.to\_csv('')

Mô tả cấu trúc :

* Cấu trúc trang web chia làm 2 phần lớn phần head và body. Nội dung chính phần body được chia thành các thẻ <div> chứa các <class=”quotes”>
* Mỗi <class=”quotes” > trong thẻ <div> chứa dữ liệu là thông tin của tác giả như tên tác giác, câu quote, các tag, link tác giả chứa trong các thẻ <a>, <span>, <small>,...



**Hình 1.2 Mô tả cấu trúc dữ liệu**

3

### 1.2. Thao tác với dữ liệu

1.2.1. Hãy đọc tất cả các thẻ html (div) với lớp là "quote" và lưu nó trong biến 'result’, hiển thị giá trị biến 'result’ ra màn hình?

Code:

result = soup.find\_all('div', class\_='quote')

#or result = soup.find\_all('div', {'class':'quote'})

# Hien thi ra man hinh

print(result)

#### 1.2.2. Hãy tìm trong biến 'result’ vừa rồi các dữ liệu có chứa nhãn "small" với class là "author" và in kết quả ra màn hình?

Code:

author = []

for quote in result:

tmp = quote.find('small', class\_='author').text

author.append(tmp)

print(author)

#### 1.2.3. Hãy viết hàm tacgiaLink() để lấy nội dung của mỗi tác giả. Với mỗi tác giả in ra màn hình các nội dung tên tác giả,đường link của tác giả, ngày tháng năm sinh và câu nói nổi tiếng của tác giả.

all\_quotes = []

def tacgiaLink():

for i in range(1, 11):

url\_page = f'https://quotes.toscrape.com/page/{i}/'

page = urlopen(url\_page)

soup = BeautifulSoup(page, 'html.parser')

quotes = soup.find\_all('div', class\_='quote')

def get\_author\_dob(link\_url):

response\_auth = requests.get(link\_url)

html\_auth = response\_auth.content

auth\_soup = BeautifulSoup(html\_auth)

auth\_tag = auth\_soup.find('span', class\_='author-born-date')

return auth\_tag.text

for quote in quotes:

#get quote

text\_quote = quote.find('span', class\_='text').text

#name author

author = quote.find('small', class\_='author').text

# get link

link = quote.find('a').get('href')

link = 'https://quotes.toscrape.com' + link

# get date of birth

dob = get\_author\_dob(link)

# tags = quote.find('div', class\_='tags').find\_all('a')

# tags\_list = []

# for tag in tags:

# tags\_list.append(tag.text)

single\_quote = [author, dob, text\_quote, link]

all\_quotes.append(single\_quote)

print(\*single\_quote, sep ="\n")

print("\n")

tacgiaLink()

4

#### 1.2.4. Hãy lưu kết quả ở câu c vào file Quote.csv tương ứng, với mỗi tác giả là 1 dòng dữ liệu. Bạn được yêu cầu thu thập ít nhất 40 câu nói nổi tiếng từ trang web trên một cách tự động theo code của các ý trên?

Code:

df = pd.DataFrame(all\_quotes, columns=['Author', 'Date of Birth', 'Quote','Link'])

df.to\_csv('Quotes.csv', index=False)

5

## Khai phá dữ liệu

### 2.1 Xử lý dữ liệu

#### 2.2.1 Một số giá trị của dữ liệu Trường ngày sinh chưa có, bạn hãy đề xuất cách điền?

Missing data là dữ liệu bị thiếu, được hiển thị như *NaN, Nat, Null, N/A,*… Missing Data xuất hiện do nhiều nguyên nhân như:

* Người dùng quên điền
* Dữ liệu bị mất trong quá trình chuyển thủ công từ cơ sở dữ liệu cũ
* Lỗi của chương trình
* Thiếu dữ liệu do trùng hợp

Trong pandas do dữ liệu có nhiều dạng và hình thức nên pandas rất linh hoạt trong việc xử lý Missing Data. Missing Data được đánh dấu mặc định là NaN để tăng tốc độ tính toán và thuận tiện. Thư viện chính dùng trong bài là pandas.

Kiểm tra các giá trị bị thiếu trong Trường ngày sinh và lấy vị trí của những vị trí bị thiếu lưu vào trong biến indexOfMissingValue :

indexOfMissingValue = df[df['Date of Birth'].isnull()].index.tolist()

print(indexOfMissingValue)

Đề xuất cách điền:

* Các tác giả đều có tên, thông tin trên trang web <https://en.wikipedia.org/wiki/Wiki>
* Đối với các tác giả thiếu giá trị chúng ta lấy tên tác giả từ vị trí được lưu trong biến indexOfMissingValue thêm vào link Wiki để tìm kiếm
* Cào dữ liệu và điền vào trường ngày sinh còn thiếu của tác giả

Code:

# columnDOB = df[['Date of Birth']]

indexOfMissingValue = df[df['Date of Birth'].isnull()].index.tolist()

print(indexOfMissingValue)

# function get date of birth on wiki

def getBirthday(i):

i = i.replace(" ","\_")

# print(i)

url = "https://en.wikipedia.org/wiki/" + i

# print(url)

page = urlopen(url)

soup = BeautifulSoup(page,'html.parser')

result = soup.find\_all('td', class\_='infobox-data')

bday = []

for i in result:

a = i.find('span')

if (a != None):

bday.append(a.text)

bday = bday[0]

return bday

for i in indexOfMissingValue:

nameAuthor = df.iloc[i]['Author']

DateOfBirth = getBirthday(nameAuthor)

df.loc[i,'Date of Birth'] = DateOfBirth

# print(columnDOB)

#### 2.2.2 Bạn hãy thêm vào Trường Tuoi (Tuổi) và đề xuất cách điền tuổi của các tác giả?

Trong trường tuổi chúng ta cần năm sinh và năm mất của tác giả

Đối với năm sinh, ta đã có dữ liệu từ trường ngày sinh của tác giả xử lý chuỗi lấy năm sinh lưu vào mảng birthYear[] :

n = df[['Date of Birth']]

a = n.values.tolist()

birthYear = []

for i in a:

# print(i)

b = str(i)[-7:-2]

birthYear.append(b)

# df.isnull()

Trường tuổi chưa có dữ liệu. Nhận thấy sau khi đăng nhập sau khi đăng nhập trang <http://quotes.toscrape.com/> đối với mỗi tác giả đều có một link tới trang <https://www.goodreads.com/> chứa thông tin đầy đủ của tác giả. Ta tiến hành cào dữ liệu về và lấy năm mất của tác giả lưu vào mảng deadYear :

Các bước cào dữ liệu:

1. Đăng nhập trang <http://quotes.toscrape.com/>

# Start the session

session = requests.Session()

# Create the payload

payload = {'username':'[admin]',

'password':'[admin]'

}

# Post the payload to the site to log in

s = session.post("http://quotes.toscrape.com/login", data=payload)

1. Lấy link tới trang goodreads của tác giả

a = i.find\_all('a')

link\_auth = a[1].get('href')

url = link\_auth

page = urlopen(url)

soup = BeautifulSoup(page,'html.parser')

1. Lấy năm sinh của tác giả trong các thẻ. Đối với tác giả chưa mất chúng ta lấy thời gian là năm của hiện tại.

## if they not die yet, get time now

if soup.find('div', itemprop="deathDate") == None:

dead = date.today().year

deadYear.append(dead)

else:

dead = soup.find('div', itemprop="deathDate").text

dead = dead.replace(' ','')[-4:]

deadYear.append(dead)

Sau khi có năm sinh và năm mất tiến hành trừ hai mảng và được trường tuổi thêm vào bảng dữ liệu tác giả:

# get age of author

age = []

for i in range(100):

result = int(deadYear[i]) - int(YearOfBirth[i])

age.append(result)

# add column age to datafarm

df['Age'] = age

### 2.2 Khám phá dữ liệu

Thư viện pandas trong python là một thư viện mã nguồn mở, hỗ trợ đắc lực trong thao tác dữ liệu. Đây cũng là bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu lẫn phát triển các ứng dụng về khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu riêng là Dataframe. Pandas cung cấp rất nhiều chức năng xử lý và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã khiến cho pandas được sử dụng rộng rãi.

Để khám phá dữ liệu trong bài này thư viện chính được dùng là pandas và vẽ biểu đồ thể hiện bằng thư viện seaborn, matplotlib.pyplot

#### 2.2.1 Thống kê về tác giả và câu nói nổi tiếng có trong bộ dữ liệu

Hàm value\_counts () được sử dụng để lấy một Chuỗi chứa số lượng các giá trị duy nhất.

Dùng value\_counts() function đếm số lần xuất hiện của tác giả chứa trong DataFrame mỗi lần xuất hiện chứa một quote

Code:

#đếm số lần tác giả xuất hiện trong dữ liệu df

df\_author = df['Author'].value\_counts()

df\_author = df\_author.reset\_index()

# print(df\_author)

df\_author.columns = ['Author', 'Quotes']

# who v/s fare barplot

sns.set(font\_scale = 0.5)

ax = sns.barplot(x = 'Author',

y = 'Quotes',

data = df\_author)

ax.set\_xticklabels(ax.get\_xticklabels(),rotation = 90)

# Show the plot

plt.savefig('TotalQuotesOfAuthor.png')

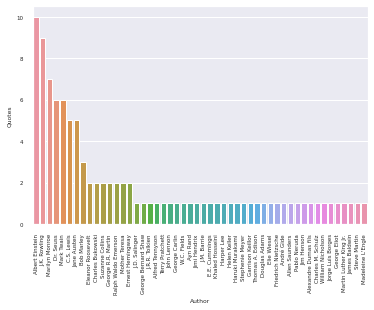
df\_author = df\_author.sort\_values(by=['Quotes'], ascending=False)

print(df\_author)

list\_author = df\_author['Author'].tolist()

print("The person with the most proverbs is: ", list\_author[0])

Biểu đồ thể hiện thống kê:



Hình 1.2 Biểu đồ quote của tác giả

Qua biểu đồ ta thấy tác giả có nhiều câu nổi tiếng nhất là Albert Einstein và J.K. Rowling đa số các tác giả còn lại có trung bình từ một đến hai câu nói nổi tiếng có trong bộ dữ liệu

#### 2.2.2 Thống kê về năm sinh và độ tuổi của các tác giả

Trong dữ liệu xuất hiện trùng lặp của tác giả trong các hàng để lấy số liệu chính xác độ tuổi và năm sinh loại bỏ các giá trị trùng lặp bằng hàm drop\_duplicates()

Thống kê năm sinh và độ tuổi tác giả bằng hàm value\_counts() và vẽ biểu đồ thể hiện

**Thống kê năm sinh**

Code :

df2 = df.copy()

# sorting by first name

df2.sort\_values("Author", inplace=True)

# dropping ALL duplicate values

df2.drop\_duplicates(subset="Author",

keep='first', inplace=True)

df\_YOB = df2['Year Of Birth'].value\_counts()

df\_YOB = df\_YOB.reset\_index()

list\_YOB = df\_YOB['index'].tolist()

# print(df\_YOB)

# df\_YOB

df\_YOB.columns = ['Year Of Birth', 'count']

# who v/s fare barplot

sns.set(font\_scale = 0.5)

ax = sns.barplot(x = 'Year Of Birth',

y = 'count',

data = df\_YOB)

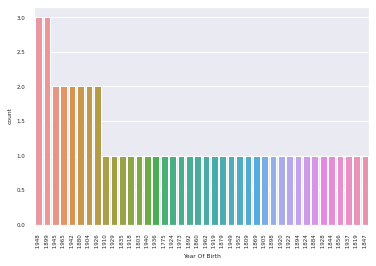
ax.set\_xticklabels(ax.get\_xticklabels(),rotation = 90)

# Show the plot

plt.savefig('YearOfBirth.png')

print('The year have most author born is ',list\_YOB[0])

Biểu đồ năm sinh:



Hình 1.3 Biểu đồ năm sinh

Qua biểu đồ dữ liệu năm tác giả sinh nhiều nhất năm

**Thống kê tuổi của tác giả**

Code:

df\_Age = df2['Age'].value\_counts()

df\_Age = df\_Age.reset\_index()

# print(df\_Age)

list\_age = df\_Age['index'].tolist()

for i in range(0, len(list\_age)):

list\_age[i] = int(list\_age[i])

average\_age = sum(list\_age)/len(list\_age)

# Age

df\_Age.columns = ['Age', 'count']

# who v/s fare barplot

sns.set(font\_scale = 0.5)

ax = sns.barplot(x = 'Age',

y = 'count',

data = df\_Age)

ax.set\_xticklabels(ax.get\_xticklabels(),rotation = 90)

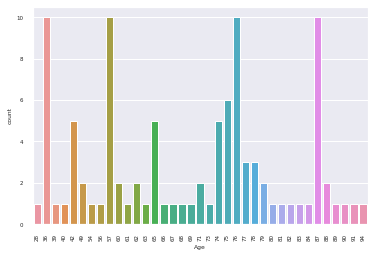
# Show the plot

plt.savefig('Age.png')

print("The age that appears the most is: ", list\_age[0])

print("The average age of the author is: ", int(average\_age))

**Biểu đồ độ tuổi của tác giả**



Hình 1.4 Biểu đồ độ tuổi của tác giả

Qua dữ liệu thống kê ta thấy có tới 40 tác giả trong độ tuổi 76, 87 , 57, 36 và trung bình tuổi tác giả 68 tuổi.

#### 2.2.3 Thống kê về các câu nói nổi tiếng như: câu dài nhất, ngắn nhất, số từ

Ta đã có cột quote trong dataFrame tách cột quote đếm số từ trong mỗi câu nói nổi tiếng bằng hàm apply và tách chuỗi split()

Thêm trường word\_count vào dataFrame tiện cho việc tính toán.

In ra màn hình câu dài nhất câu ngắn của tác giả.

Code:

df2 = df.copy()

# data.insert(5,"Count\_Words", data["Quote"].apply(lambda x: len(str(x).split(' '))), True)

words\_count = df2["Quote"].apply(lambda x: len(str(x).split(' ')))

print(words\_count)

df2['Count\_Words'] = words\_count

df2 = df2.sort\_values(by=['Count\_Words'], ascending=False)

# df2.head(10)

# df2 = df2.reset\_index()

df\_count\_word = df2[['Author','Quote','Count\_Words']]

lengthest\_word = df\_count\_word.iloc[0]

sortest\_word = df\_count\_word.iloc[-1:]

# print('Lengthest sentence:')

# print(lengthest\_word)

# print('\nsortest sentence:')

# print(sortest\_word)

**Thống kê về các từ được sử dụng trong các câu nói**

Chúng ta đưa hết tất cả các câu nổi tiếng của tác giả vào một chuỗi để xử lý

Làm sạch dữ liệu, loại bỏ các ký hiệu không cần thiết có trong câu để tiện lợi trong việc xử lý dữ liệu

all\_quotes = ''

for quote in quotes:

all\_quotes += quote

all\_quotes = re.sub(r'[^\w]', ' ', all\_quotes)

all\_quotes = re.sub(r' +', ' ', all\_quotes)

Trong dữ liệu còn các stop word loại bỏ chúng bằng thư viện wordCloud và thể hiện tần suất xuất hiện các từ bằng value\_counts() function:

Code:

from collections import Counter

df2 = df.copy()

quotes = df2['Quote']

all\_quotes = ''

for quote in quotes:

all\_quotes += quote

all\_quotes = re.sub(r'[^\w]', ' ', all\_quotes)

all\_quotes = re.sub(r' +', ' ', all\_quotes)

all\_quotes = all\_quotes.lower()

stop\_words = stopwords = set(STOPWORDS)

word\_tokens = word\_tokenize(all\_quotes)

list\_words = [w for w in word\_tokens if not w.lower() in stop\_words]

list\_words = []

for w in word\_tokens:

if w not in stop\_words:

list\_words.append(w)

df3 = pd.DataFrame (list\_words, columns = ['word'])

rank\_words = df3['word'].value\_counts()

rank\_words.head(10)

Biểu đồ thể hiện tần suất của các từ bằng wordCloud:

Code:

comment\_words = ''

stopwords = set(STOPWORDS)

# iterate through the csv file

for val in df2['Quote']:

# typecaste each val to string

val = str(val)

# split the value

tokens = val.split()

# Converts each token into lowercase

for i in range(len(tokens)):

tokens[i] = tokens[i].lower()

comment\_words += " ".join(tokens)+" "

# filtered\_words = [w for w in allwords if not w in stopwords]

wordcloud = WordCloud(width = 800, height = 800,

background\_color ='white',

stopwords = stopwords,

min\_font\_size = 10).generate(comment\_words)

# print(wordcloud.words\_)

# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None)

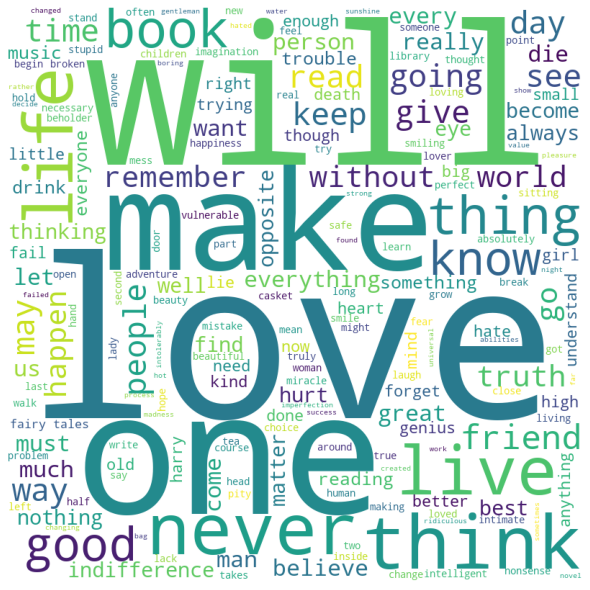
plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

wordcloud.to\_file("count\_words.png")

Output: 

Qua thống kê dữ liệu các từ có tần suất xuất hiện nhiều nhất các câu nói nổi tiếng của tác giả là “love”,”'will”, “one”, “make”, “think”, “never”

### **2.3.** **Trích xuất đặc trưng- Feature Extraction**

**Hướng tiếp cận: sử dụng thư viện CountVectorizer**

CountVectorizer sử dụng dữ liệu văn bản cho mô hình dự đoán, văn bản phải được phân tích cú pháp để loại bỏ một số từ nhất định – quá trình này được gọi là mã hóa. Sau đó, những từ này cần được mã hóa dưới dạng số nguyên hoặc giá trị dấu phẩy động, để sử dụng làm đầu vào trong thuật toán học máy. Quá trình này được gọi là trích xuất đặc trưng (hoặc vectơ hóa) .

Scikit-learning CountVectorizer được sử dụng để chuyển đổi một bộ sưu tập các tài liệu văn bản thành một vectơ có số lượng thuật ngữ / mã thông báo. Nó cũng cho phép xử lý trước dữ liệu văn bản trước khi tạo biểu diễn vectơ. Chức năng này làm cho nó trở thành một mô-đun biểu diễn tính năng rất linh hoạt cho văn bản.

Mô hình được sử dụng trong bài là mô hình Vector Space hay còn gọi là mô hình Bag-of-Words (BoW).

**Bag-of-Words** là một cách tiếp cận rất trực quan cho vấn đề này, các phương pháp bao gồm:

1. Tách tài liệu thành các token theo một số kiểu mẫu.
2. Chỉ định trọng số cho mỗi mã thông báo tỷ lệ với tần suất mà nó hiển thị trong tài liệu và / hoặc kho tài liệu.
3. Tạo ma trận thuật ngữ tài liệu với mỗi hàng đại diện cho tài liệu và mỗi cột địa chỉ mã thông báo

Các đối tượng vectorizer do Scikit-Learn cung cấp khá đáng tin cậy ngay từ đầu, chúng cho phép chúng ta thực hiện tất cả các bước trên cùng một lúc một cách hiệu quả và thậm chí áp dụng tiền xử lý và các quy tắc liên quan đến số lượng và tần suất của token.

Code:

#2.3

df2 = df.copy()

authors = df2['Author'].tolist()

# create a dataframe from a word matrix

def wm2df(wm, feat\_names):

# create an index for each row

doc = ['Doc{:d}'.format(idx) for idx, \_ in enumerate(wm)]

df = pd.DataFrame(data=wm.toarray(), index=doc,columns=feat\_names)

return(df)

# instantiate the vectorizer object

cvec = CountVectorizer()

# convert the documents into a document-term matrix

wm = cvec.fit\_transform(df2['Quote'])

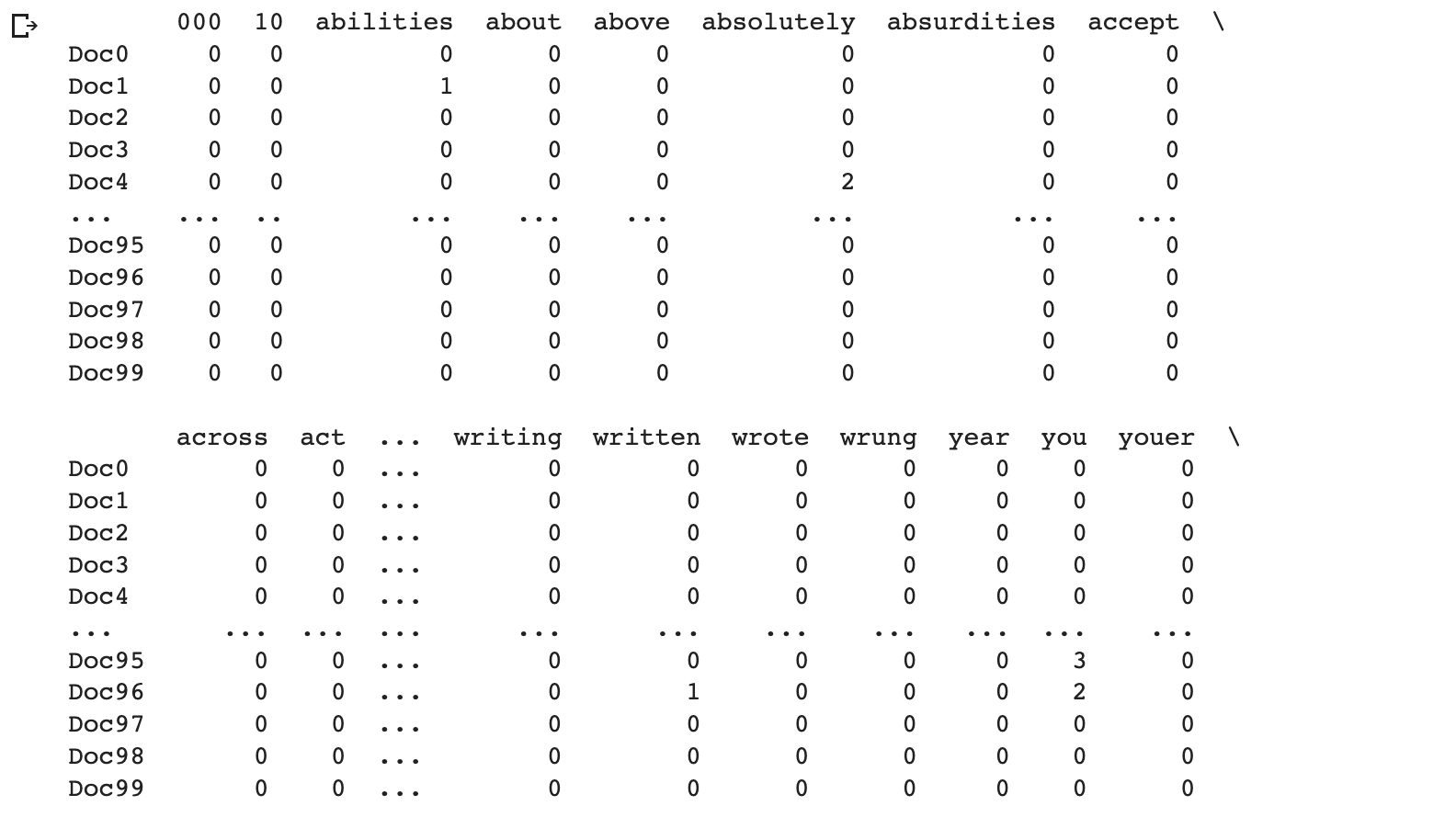
# retrieve the terms found in the corpora

tokens = cvec.get\_feature\_names()

# create a dataframe from the matrix

wm2df = wm2df(wm, tokens)

Output:



### 2.4. Suy luận

#### 2.4.1 Hãy dự đoán tên của người nổi tiếng theo câu nói dựa trên các đặc trưng bạn trích xuất ở trên và đánh giá trên bộ dữ liệu đã cho với tỉ lệ Train/Test và các độ đo phù hợp?

Giới thiệu Naive Bayes là một trong những thuật toán đơn giản nhất nhưng mạnh nhất để phân loại dựa trên Định lý Bayes với giả định độc lập giữa các yếu tố dự đoán. Mô hình Naive Bayes rất dễ xây dựng và đặc biệt hữu ích cho các tập dữ liệu rất lớn. Có hai phần của thuật toán này:

* Naive
* Bayes

Trình phân loại Naive Bayes giả định rằng sự hiện diện của một tính năng trong một lớp không liên quan đến bất kỳ tính năng nào khác. Ngay cả khi các tính năng này phụ thuộc vào nhau hoặc phụ thuộc vào sự tồn tại của các tính năng khác, tất cả các tính chất này đều góp phần độc lập vào xác suất rằng một loại trái cây cụ thể là táo hoặc cam hoặc chuối, và đó là lý do tại sao nó được gọi là "Naive . "

Trong thư viện Scikit-Learn có hỗ trợ 4 dạng Naive Bayes (NB): Gaussian NB, Multinomial NB, Complement NB, Bernoulli NB

Trong bài dự đoán tác giả dựa theo câu nói ở đây sử dụng Multinomial NB:

* Multinomial NB: Sử dụng phương pháp đếm tần suất (frequency counting) như trong bài toán trường học ở bài lý thuyết để tính khả năng (likelihood). Thích hợp với các tập dữ liệu (dataset) có dữ liệu dạng phân loại (categorical value).

Code:

authors = df2['Author'].tolist()

# chia tập dữ liệu train và test tỉ lệ 80/20

features\_train, features\_test, labels\_train, labels\_test = train\_test\_split(wm, authors, test\_size=0.2, random\_state=75)

# print(features\_test)

# Khởi tạo Multinomial NB model

nb = MultinomialNB()

# Fit tập dữ tập liệu huấn luyện (training set) với Multinomial NB model vừa tạo

nb.fit(features\_train, labels\_train)

# kiểm tra độ chính xác

score = accuracy\_score(labels\_test,nb.predict(features\_test))

print("score: ",score)

# Chuyển đổi một tập hợp các tài liệu văn bản thành một ma trận số lượng mã tokens.

text = ["It is our choices, Harry, that show what we truly are, far more than our abilities."]

transformed = cvec.transform(text)

# nb.predict(transformed) chứa các giá trị số tương ứng với danh mục.

print('The author is:' ,nb.predict(transformed))

Output:

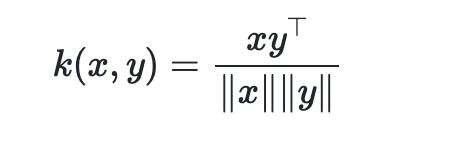
score: 0.1

The author is: ['J.K. Rowling']

#### 2.4.2 Hãy đề xuất cách tính độ tương đồng phong cách nói giữa các tác giả và tìm ra các tác giả có phong cách nói tương đồng nhau nhất?

Giới thiệu thư viện sử dụng trong bài:

* SentenceTransformer('bert-base-nli-mean-tokens'): đây là một mô hình biến đổi câu: Nó ánh xạ các câu và đoạn văn tới một không gian vectơ dày đặc 768 chiều và có thể được sử dụng cho các tác vụ như phân cụm hoặc tìm kiếm ngữ nghĩa.
* Cosine\_similarity tính tích số chấm chuẩn hóa L2 của các vectơ. Đó là, nếu x và y là các vectơ hàng, tính tương tự cosin của chúng được định nghĩa là:



điều này được gọi là tương tự cosin, bởi vì chuẩn hóa Euclidean (L2) chiếu các vectơ lên ​​hình cầu đơn vị, và tích chấm của chúng sau đó là cosin của góc giữa các điểm được ký hiệu bởi các vectơ.

Cách tiếp cận:

1. Khởi tạo dữ liệu cột author và quote dạng dictionary.
2. Mã hoá các câu nói bằng SentenceTransformer('bert-base-nli-mean-tokens')
3. Chuẩn bị dữ liệu xong tiến hành so sánh độ tương đồng các câu nói.
4. Sắp xếp độ tương đồ từ cao đến thấp của các giả.

Code:

dict\_df = {}

list\_Of\_author = df2['Author'].unique().tolist()

# Khởi tạo dữ liệu kiểu dictionary

for author in list\_Of\_author:

tmp\_df = df2.loc[df2['Author'] == author]

key = author

dict\_df.setdefault(key,[])

tmp\_df.reset\_index()

list\_quotes = tmp\_df['Quote'].tolist()

string = ' '.join(list\_quotes)

dict\_df[key] = string

key\_list\_author = list(dict\_df.keys())

val\_list\_quote = list(dict\_df.values())

# print(dict\_df)

# model nhúng câu, văn bản và hình ảnh.

model\_ST = SentenceTransformer('bert-base-nli-mean-tokens')

#Các câu được mã hóa bằng cách gọi model\_ST.encode ()

quote\_endcode = model\_ST.encode(val\_list\_quote)

# print(quote\_endcode.shape)

# print(quote\_endcode)

for j in range(len(quote\_endcode)):

temp\_quote\_endcode = quote\_endcode[np.arange(len(quote\_endcode))!=j]

#so sánh 1 câu với các câu nói khác

similarity = cosine\_similarity([quote\_endcode[j]], temp\_quote\_endcode)

list\_key = key\_list\_author.copy()

list\_key.pop(j)

#sắp xếp độ tương đồng giữa các tác giả với nhau

for n in range(len(similarity[0])-1, 0, -1):

for i in range(n):

if(similarity[0][i] < similarity[0][i+1]):

similarity[0][i], similarity[0][i+1] = similarity[0][i+1], similarity[0][i]

list\_key[i], list\_key[i+1] = list\_key[i+1], list\_key[i]

# print(list\_key)

print("Authors with the highest similarity to ",key\_list\_author[j], 'respectively are:', list\_key)

Ouput:



Cosine similarity là một trong những cách tốt nhất để đánh giá hoặc đo lường sự giống nhau giữa các tài liệu. Không phân biệt kích thước, công cụ đo độ tương đồng này hoạt động tốt. Chúng ta cũng có thể thực hiện điều này mà không cần mô-đun sklearn. Nhưng nó sẽ là một nhiệm vụ tẻ nhạt hơn. Sklearn đơn giản hóa điều này.

9

## TÀI LIỆU THAM KHẢO

1. <https://en.wikipedia.org/wiki/Wiki>
2. <https://www.w3schools.com/python/>
3. <https://chickgolden.com/countvectorizer-la-gi-1637494647>
4. <https://huggingface.co/sentence-transformers>
5. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine\_similarity.html